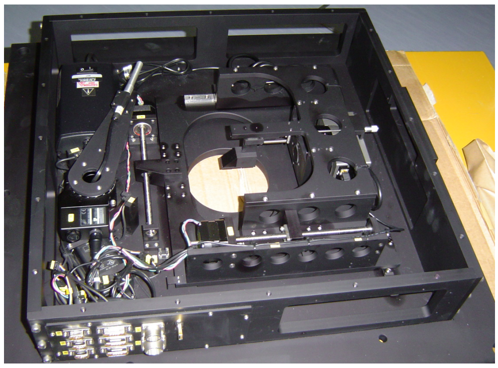
かなた望遠鏡用オートガイダー
及びガイド星自動捜索ソフトウェアの開発

広島大学理学部物理科学科
高エネルギー宇宙・可視赤外線天文学研究室
B034188 佐藤久之
主査 川端弘治　副査 高橋徹

2011 年 2 月 25 日

概 要

広島大学かなた望遠鏡は，2006 年に建設されて以来，性能の良さと豊富な観測時間を活かして，様々な観測研究に用いられ，成果を挙げてきた．HOWPol はかなた望遠鏡専用に開発された観測機器であり，2009 年以降，可視域の撮像・観測や偏光観測に用いられている．最近は超新星などの分光観測の比率が増しているが，分光観測では 2.3 秒角幅という狭いスリットを用いて観測を行うため，追尾誤差による光の遮蔽が発生し，観測効率の低下が問題となっていた．



オートガイダー内部の様子

本研究では，追尾誤差をリアルタイムで補正するオートガイダーの整備を行った．これは，観測視野内に写りこんでいる星の一つをガイド星としてモニタリングし，その位置ずれを検出して望遠鏡にフィードバックを掛けるものである．私は，ハードウェアは既に完成していたものの実用可まで至らなかった，HOWPol 用のオートガイダーを制御するソフトウェアを開発した．具体的には，望遠鏡を天体へ向けた直後に自動的にオートガイドを開始するため，観測天体に応じた恒星カタログからのガイド星候補の抽出・決定，XY ステージ駆動によるオートガイダー CCD への導入，ガイド星重心ズレの望遠鏡へのフィードバックの一連手順を自動で行えるように整備した．これにより分光観測などの観測効率が大幅に向上すると期待される．

目 次

図 目 次

表 目 次

第1章　序論

1.1　かなた望遠鏡

広島大学宇宙科学センター附属東広島天文台は，広島大学東広島キャンパスから車で約20分程度の距離にある福成寺付近の山中に建設された，アクセスの良い観測施設である．かなた望遠鏡は，2006年に国立天文台からこの東広島天文台へ移設された主鏡口径1.5 mの可視近赤外望遠鏡である．この望遠鏡は，国立天文台ハワイ観測所にある大型光学赤外望遠鏡「すばる」で使用する観測装置の開発，評価をする目的で国立天文台に建設されたものであるが，すばるの第一観測装置開発が全て終了した後，本格的な天文，宇宙研究への有効な活用を提案した広島大学へ移管された．

かなた望遠鏡を用いた主な研究目的には，ガンマ線バースト（GRB）の即時観測による高エネルギー宇宙現象の解明や，種々の超新星の密な早期観測などがある．

GRBの即時観測にはアラート対応観測が含まれる．GRB等の突発天体を検出した時に，インターネットを通じてアラートを受信し，天体の座標を取得後，即時観測に移行するものである．かなた望遠鏡はGRBの早期観測を実現するため，GRB発生後60秒程度で自動的に目的天体に向ける体制が整えられている．これは主鏡口径1.5 mクラスの望遠鏡では世界最高水準の速度である．

また超新星の観測においては，かなた望遠鏡は測光と分光観測を行っている．主に超新星爆発直後の早期の段階から観測を行い，すばる望遠鏡の後期観測と合わせるなどして超新星現象の詳細な解明が期待できる．

1.2　一露出型偏光観測装置HOWPol

HOWPol（Hiroshima One-shot Wide-field Polarimeter）は，広島大学宇宙科学センターで開発を進めている，突発天体に特化した偏光観測装置である．全長はおよそ1 m，重量約200 kgの装置であり，かなた望遠鏡のナスミス焦点に常設されている．本格的な偏光装置は通常，光軸が非対称であるナスミス焦点に取り付けることはないが，突発天体の観測に伴う常設の必要性から，本装置はあえてナスミス焦点に取り付けてある．将来の大型望遠鏡ではナスミス焦点のみ実装されるものもある．その意味でもナスミス焦点に取り付けた．偏光装置がどれほどの性能を発揮できるか注目されている．



図 1.1: 東広島天文台 かなた望遠鏡

表 1.1: かなた望遠鏡仕様

項目	仕様
光学系	リッチー・クレチアン光学系
主鏡	有効口径 1540 mm/主鏡の F 比 = 2.0
焦点モード	カセグレン焦点・ナスミス焦点とも F/12.01
焦点面スケール	カセグレン・ナスミスとも 11.15 秒角/mm
分解能	1”FWHM
最大駆動速度	方位軸 5°/sec
	高度軸 2°/sec
最大加速度	1°/sec^2 以上
架台	経緯台

HOWPol の検出器は，浜松ホトニクスで開発が進められた完全空乏型 CCD である．このCCD は長波長域での感度に優れ，1000-1100 mm では従来の CCD の 2 倍以上の感度を持っていることが特徴である．また，HOWPol のモードには測光，偏光の他に，幅 2.3 秒角のスリットを焦点マスクに用いた分光モードがある．表 1.2 に，HOWPol の基本仕様を明記しておく．

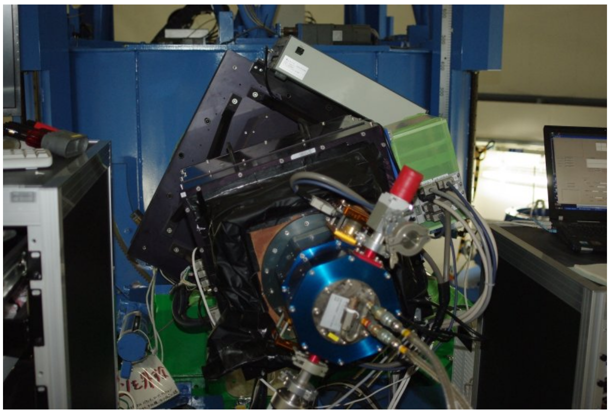


図 1.2: HOWPol

表 1.2: HOWPol の仕様

項目	仕様
視野	15 分角（56 %縮小光学系）
検出器	完全空乏型 CCD 2k × 4k × 2
ピクセルスケール	0.3 秒角/pix
波長領域	4500-11000 Å
フィルタ	広帯域（B, V, Rc, Ic, z'），各種狭帯域
モード	撮像，偏光，分光
限界等級	測光モード 19.8 mag（10 min exp, S/N = 50）
	偏光モード 15.9 mag（10 min exp，δP = 0.2 %）

1.3　オートガイダー

かなた望遠鏡は恒星の日周運動に応じた追尾機能を有しているが，長時間露光しているとわずかにずれを生じる．これを追尾後差という．

特に分光観測では長時間露光を必要とするが，分光観測に用いるスリット幅は 2.3 秒角と狭く，星の位置がずれると光が遮蔽され，光量を失ってしてしまうことになる．このため現状の観測では，1 枚撮影（3-5 分露出）ごとに観測を中断し，望遠鏡の方向を修正することで追尾誤差を修正している．しかし，この方式では非常に手間がかかり，観測時間も制約されてしまう．

このようなずれを自動で，リアルタイムに検知，修正し，観測における効率を改善するのがオートガイダー（自動追尾装置）である．

本オートガイダーは HOWPol の内部に組み込まれている．ガイドする参照星（オフセットガイド星）の光を導く部品が HOWPol の視野を遮るため，観測天体の邪魔にならないようにガイド星を選択する必要がある（図 1.3）．このオフセットガイド星の輝度重心が常に同じ位置になるように望遠鏡に追尾誤差の修正を送り続ける．

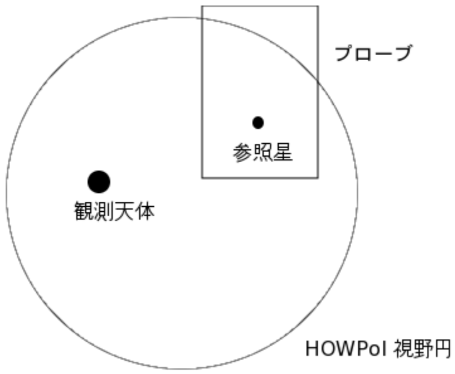


図 1.3: オートガイダー使用時の HOWPol の視野

1.4 HOWPol用オートガイダーの構成

HOWPol のオートガイダーは，前任者の平木一至氏（広島大学大学院理学研究科 D1）が開発を進めたもので，以下のような構成になっている．

1.4.1 検出部

オートガイダーの検出部は，導入用 20×20 mm 45 ° 直角プリズム（シグマ光機 RPB2-20-550）と，BITRAN 製冷却 CCD カメラ BS-41L から成る．

このカメラは有効画素数 1360×1024 画素，受光面積 8.8×6.45 mm（かなた望遠鏡に取り付けた時の視野は 97.8×73.6 秒角）であり，USB，および PCI 通信によってノート PC からの CCD 制御を可能としている．本研究では USB 通信を利用している．



図 1.4: ビットラン社製 BS-41L

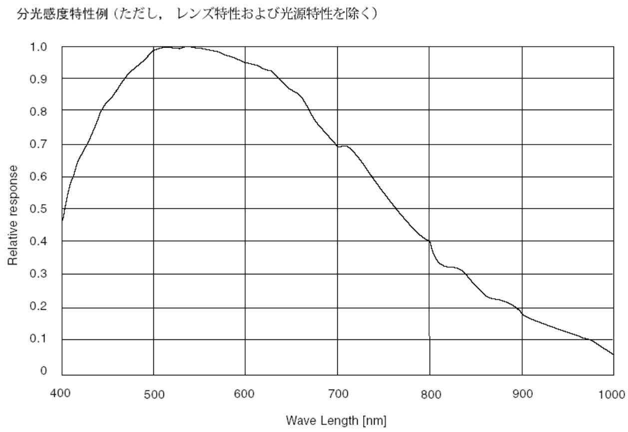


図 1.5: BS-41L 分光感度特性

表 1.3: BS-41L 仕様表

項目	仕様
メーカー/製品名	ビットラン/BS-41L
インターフェース	USB or PCI
画像転送速度	0.8 秒 （USB1.1 の場合は 4 秒）
CCD 素子	モノクロ
ピクセル数	1360 × 1024
冷却機能	ペルチェ冷却 （水冷機構付き）
冷却温度	空冷時 外気温-25°C
A/D 変換	16 bit/8 bit
露光時間	1 ms から 1 hour
受光面積	8.8 × 6.45 mm （97.8 × 73.6 秒角）
ピクセルサイズ	6.45 × 6.45 μ m

本オートガイダーでは露出時間は 0.001 秒から 6553 秒の間で自由に設定できる．星の明るさや観測時の天候など観測条件によって露出時間を変更する必要があり，その時間はこの範囲内で設定することになる．

またビニングは任意の設定が可能であるが，4×4 で固定されている．これは CCD 上における恒星像のピクセルサイズを考慮している．かなた望遠鏡の焦点距離は 18501.7 mm，オートガイダー用 CCD のピクセルサイズは 6.45 μm であるから，1 ピクセルあたりの見込み角は

$$\theta = \arctan\left(\frac{0.00645}{18501.7}\right) \simeq 0.072''$$

である．天体は一般に無限遠にあると考えて良いが，実際は地球大気のゆらぎにより有限の大きさを持つ．この恒星像の広がり（半値幅 FWHM）をシーイングサイズと言う．東広島天文台では良好な条件の場合，シーイングサイズは 1-2 秒角程度なので，ビニングなしだと恒星像は 14×14 ピクセル程度の大きさになる．これでは解像度が無駄に高い上に，データの読み出しに時間がかかり，なるべく早い周期で位置ずれの修正を行う場合には不利になる．しかし，一方でシーイングサイズが 1×1 ピクセル程度になると位置や測光の誤差が大きくなってしまう．そこで，本オートガイダーではナイキスト・サンプリング定理を考慮し，かつ HOWPol に使用されている CCD のピクセルスケールと同程度となる設定である 4×4 ビンニングが採用されている．

1.4.2　駆動・制御系

オートガイダーの筐体には，CCD カメラとピックアップミラーを二次元で駆動させる X，Y ステージと，分光観測用のコンパリソンランプ，またコンパリソンランプを駆動させる θ ステージが搭載されている．X，Y，θ ステージはそれぞれ 1 つのモータと 2 つのリミットセンサ（フォトインタラプタ）を持つ．それぞれのモータはコンテック社 AL シリーズ対応 2 軸ステッピングモータドライバ内蔵コントローラスレイブ CD-773/ADB5331A を介して PC から送られたコマンドによって駆動する．

平木氏によるソフトウェアではステージをマニュアルで動かす機能は実装してあったが，任意の天体が入るようにステージを導入する機能や．自動的にオートガイドを行う機能は未実装であった．

1.5　オートガイダーに要求される性能

超新星など分光を行う場合は，多くは 14-16 等の明るさしかないため，良い S/N 比をもつスペクトルを取得するには数十分以上の長時間の観測が必要となる．しかし，望遠鏡の追尾誤差の影響で 5-10 分おきに天体をスリット上に入れ直す作業（数分間）が必要とな

表 1.4: モータおよびステージの仕様

項目	X 軸	Y 軸	θ 軸
メーカー	KSS/THK	KSS/THK	シグマ光機
型式	TS3667N14E2/SRS12M	TS3667N14E2/SRS12M	SGSP-60YAW-0B
1 パルス移動量	4 μm（0.031 pixel）	4 μm（0.031 pixel）	0.005°
送り方式	カップリングレス	カップリングレス	ウォーム
	ボールねじ式	ボールねじ式	ウォームホイール
ねじ軸外径	4 mm	4 mm	-
通信方式	USB	USB	USB
ストローク	115 mm	115 mm	-
駆動速度	最高速度 11.06 mm/s	最高速度 11.06 mm/s	-
	駆動開始速度 0.66 mm/s	駆動開始速度 0.66 mm/s	-

り，効率が悪い．そこで，オートガイダーを導入し，自動追尾のズレの補正を行うことによって限界等級を深めることを目指したい．オートガイダーの導入によって観測可能なターゲットは増え，重要なデータを取得できる機会も多くなるはずである．

かなた望遠鏡に取り付けられているスリットの幅は 2.3 秒角であるため，そのおよそ 4 分の 1，約 0.6 秒角より大きくずれると情報の損失が顕著になる．本プログラムはこの 0.6 秒角未満という追尾精度を目標として開発を進める．また，効率の良い観測を行うためには，オートガイド開始までの作業をなるべく自動化して，手際良く行えるシステムにする必要がある．私はガイド星の自動探索を含む様々な処理を自動化することも目指して，実用に耐えるプログラムの開発を行った．

第2章　オートガイダーソフトウェアの開発

本研究のオートガイダーソフトウェアはVisual C++環境で開発した．Visual C++を用いた理由は，ガイド用CCDカメラ，及びモータ制御のためのライブラリとしてVisual C++用のものだけが提供されているためである．また，ソフトウェアの2008年に前任者の平木氏が開発を進めており，その設定を一部引き継いでいることも理由である．本研究ではオートガイドを自動的に行えるようにするためのプログラムの開発を目指した．ハードウェアは2008年10月までに完成しているが，制御ソフトについては§1.4.2でも述べたように未完のままであった．

2.1　観測の実際と開発目標

HOWPolにおける一般的な分光観測の手順と，それに要する典型的な時間は

1. 撮像モードで試し撮り（1.5分）
2. 分光したい星を選定（0.3分）
3. 光路中の焦点面にスリットを入れ，試し撮り（1分）
4. 分光したい星を入れたいスリット上の場所を選定（0.5分）
5. スリット上に観測天体が入っているか確認するため試し撮り（0.5分）
6. 入っている場合はグリズムを入れて1枚分光撮像（6分）
7. 撮りたい枚数だけ1から7を繰り返す

である．この分光観測手順は，一度行えば完了という訳ではなく，追尾誤差があるため露出のたびに手順を繰り返さなくてはならない．つまり，繰り返し行わないと，スリットから星が逃げてしまい，光量の損失が生じる．観測時間の損失においても，観測1回の繰り返しで分光撮像までに約4分を要し，5枚撮像する場合は1天体につき約16分の時間の損

失が生じることになる．その都度観測者も自身の作業を中断し，HOWPol 制御用 PC の前に向かわなければならず，非常に手間である．オートガイドを導入することで時間の損失も改善されることが期待される．

また，オートガイド開始に掛かる時間をなるべく短くすることも重要である．本研究ではオートガイダーソフトウェア全体を完成させるために，短時間でオートガイド開始できるプログラムの開発も進めた．

2.2　サーバーモードとマニュアルモード

本プログラムでは，サーバモードとマニュアルモードの 2 つのモードを実装する．基本的にはサーバモードで常駐プログラムとして動かし，望遠鏡を観測天体に向けた段階からガイド星の選定，オートガイドを行う．曇天になってガイドが不可になった場合や，意図しない挙動になった場合はガイドをストップする命令を送り，いつでもオートガイドの開始，停止ができるものにする．一方，マニュアルモードは，ガイド星を観測者が選定するところに特徴がある．観測天体付近のなるべく広い視野を撮像したい場合など，ガイド星として使う領域を限定したときにこのモードを使うことを想定している．ガイド星を観測者自身が選びたい場合にこのモードに移行し，ガイド星を選定後，再びオートガイドを開始する．

2.3　恒星カタログからのデータ抽出

2.3.1　恒星カタログ

オートガイダーでガイドする恒星（ガイド星）を選定する際，ガイド星の自動導入や露出時間の決定のために，ガイド星の赤経，赤緯，等級の情報が必要となる．このような情報は，望遠鏡の指向位置に応じて恒星カタログから求める．広く用いられている恒星カタログには Guide Star Catalog（GSC），USNO，Hipparcos/Tycho などがあり，それぞれの特徴は表 2.1 のようになっている．

オートガイダーで恒星カタログを利用する場合，位置の精測を行わないことから高い位置精度は必要としない．また，暗すぎる星はガイド星として適さないため 15 等程度までの星が収録されていれば充分である．そこで本オートガイダーでは GSC1.2 を利用することにした．

表 2.1: 恒星カタログの比較

カタログ名	星数	最微等級	位置精度	備考
GSC1.2	約 1900 万	16 等	0.5 arcsec	ハッブル宇宙望遠鏡姿勢制御用
USNO-A2.0	約 5 億	約 20 等	0.25 arcsec	膨大な星の数と高い位置精度
Hipparcos/Tycho	約 100 万	約 12 等	0.03 arcsec	最高の位置精度

2.3.2 Guide Star Catalog からのデータ抽出

GSC（http://archive.stsci.edu/gsc/より入手可能）はバイナリ形式で配布されているため，扱いやすさや視認性を高めるためにバイナリ形式からテキスト形式へのデコードを行った．ただし，それぞれの天体にカタログでの ID，赤経，赤緯，位置誤差，等級，等級誤差，天体の種類などのデータが含まれており，全てを含めると非常に大きな容量が必要で，処理に時間がかかってしまう．そこで GSC の必要な情報のみを抽出するようにした．

抽出条件は以下の通りである．

1. ガイド星として使用するのに最低限必要なデータである赤経・赤緯・等級のみを抽出．
2. 東広島の地理的条件より赤緯-45° より南側のデータを省く．
3. 銀河など恒星ではないデータは省く．
4. GSC は固有運動が考慮されておらず，同じ天体（ID）だが元期の異なる乾板のデータも収録されているので，2 つ目以降の同 ID 天体は省く．

まずはバイナリデータの把握から行った．GSC では 1 つの天体に対して 96 ビット（12 バイト）が割り当てられており，収録されている情報と，各情報に与えられているビット数が公開されている．ただし，情報の順序関係は不明であったため，収録されている情報と対応する上位からのビット数との関係が表 2.2 の通りであることを見出した．

バイナリデータからデコーディングする際は，それぞれのファイルのヘッダにテキストで書かれてある，星数，赤経・赤緯それぞれ小さい方の境界値（オフセット），等級のオフセット，赤経・赤緯・等級のスケーリングファクターが必要となる．バイナリより抜き出した赤経・赤緯・等級は，それぞれスケーリングファクターで除算した後，オフセットを加える．赤経赤緯の単位は度である．

このようにして，データを一つのファイルにまとめた後，天体情報の検索速度を上げるために，赤緯範囲に応じて 113 個のファイルに分割した．各ファイルには，観測天体と同

表 2.2: GSC レコードに収録されている情報

GSC field	bits
spare	1
GSC-ID	2 - 15
RA	16 - 37
DEC	38 - 56
pos-error	57 - 65
mag-error	66 - 72
magnitude	73 - 83
mag-band	84 - 87
multiple	88
spare	89
class.[0=star; 3=non-stellar]	90 - 92
plate-id	93 - 96

一の視野内（8'×8'）に入る星をすばやく捜し出せるように，境界線を 8' だけ広げた領域の星を加えるようにした．また，C 言語の三角関数に用いやすいように赤経・赤緯の単位を度からラジアンに直した．この分割による速度向上については §3.2 で述べる．

2.4　オートガイダーソフトウェアの流れ

本セクションではサーバタイプの大まかな手順を解説する．まずサーバタイプの全手順を述べた後，マニュアルモードとの切り替えについて述べる．

サーバタイプではオートガイダー全手順を自動で行えるようにする．その手順は以下の通りである．

1. 望遠鏡の現在の指向情報を得る
2. 視野中心の赤経赤緯から半径 8 分円内の恒星情報を検索する
3. 得た恒星情報の中からガイド星を選ぶ
4. オートガイダーステージを動かしてガイド星がオートガイダー視野中心に来るようにする

5. ガイド星の等級からオートガイダーの露出時間を設定する
6. オートガイダー CCD で 1 枚撮る
7. ガイド星の重心位置を求める
8. 再び 1 枚撮る
9. 重心位置を求め，前回の位置と比較しズレを求める
10. ズレ量を赤経赤緯値に変換し，修正する方向にかなた望遠鏡の向きを移動する
11. 8. - 10. の繰り返し

次節で述べるが，適切なガイド星が見つからない場合はマニュアルモードに移行し，観測者自身がガイド星を選ぶようにする．ガイド星を選んだ後は同じフローチャートを進む．また，薄雲がかかってガイド星のカウントが低下した時にもマニュアルモードに入り，観測者の命令によって露出時間を長くしたり，ガイド星を変えたりする機能も必要になる．

2.5　ガイド星の選定

オートガイダー使用時には，ピックアップミラーを内包するプローブによる影が観測視野の一部を遮蔽するため（図 2.1），ガイド星は観測天体を遮蔽しないように観測天体から離れた場所にある恒星を選択する必要がある．同時に，なるべく早く望遠鏡へフィードバックを行うためにガイド星は明るい星でなければならない．ただし，明るすぎるとサチュレートを起こしやすくなるため，適切な明るさの範囲を定める必要もある．

そこでガイド星の位置と明るさをそれぞれ点数化した関数を考え，最も値が大きい星をガイド星候補として採用することとした．ガイド星の位置に関しては，観測天体から遠い場所にある星ほど点数を高くし，明るさに関しては，定められた明るさの範囲内で明るい星ほど点数を高くする．

点数には一定の基準を定め，それよりも高い点数の星が無い場合，あるいは測光に使うための天体にプローブの影が入って困る場合は，マニュアルモードに移り，観測者がガイド星を選ぶようにする．

ガイド星選定可能領域についてもう少し詳しく見ていく．HOWPol の検出部は CCD が 2 枚使われており，半径 8 分の視野円は左半分（Chip1）と右半分（Chip0）に分割されている．現在の観測では，Chip1 の中心付近に観測天体を導入して観測することが多いため，Chip0 でガイド星を探すのが良い場合がほとんどである．ただし，プローブの影が図

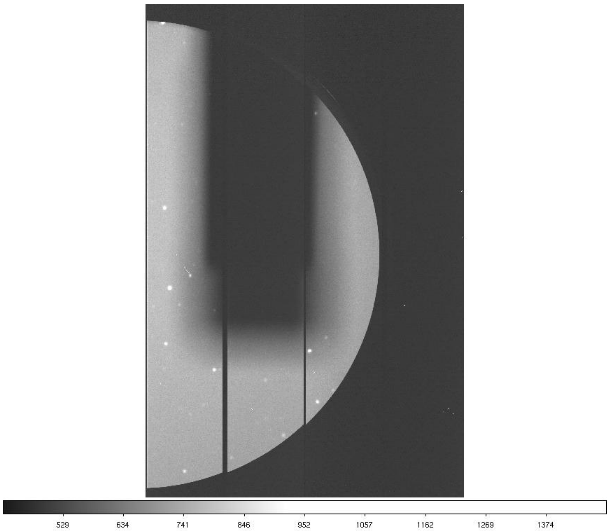


図 2.1: オートガイダー使用時の観測視野

2.1 のように約 5.3 分角の幅を持つために，Chip0 の天体をガイド星候補とする場合にも，Chip1 の天体を遮蔽する場合があることに注意する必要がある．プローブの設計値と F12 の収束光から計算される影は約 6.2 分角幅であるが，周辺部分では淡くなるため測定された値は妥当なものと思われる．観測天体がプローブの影響を全く受けない領域は 6.2 分角幅であると考えると，Chip1 側から 3.1 分角以内に入る部分はガイド星として使えない場合がある．

また，先にも述べたように観測天体よりなるべく遠いところに存在する星をガイド星に選ぶのが望ましいが，Chip1 の下側の星を選んでしまうとプローブの影により Chip1 がほぼ全体的に隠されてしまうので，Chip1 では上側のみガイド星選定可能領域となる．これらよりプローブ移動禁止禁止領域の概略は図 2.2 のようになる．この禁止領域内にある天体はガイド星候補としない．

ガイド星候補の明るさに関しては，まず 12 等級から 5 等級までの範囲とし，明るい星ほど優先度が高いものとする．12 等級より暗いものを省く理由としては，オートガイダー CCD での露出時間が長くなり，追尾誤差 0.6 秒以下という目標が達成されなくなることが予想されるためである．5 等級より明るい星を省くのはサチュレートを起こしやすくなる可能性があるためである．実際には星の等級と必要な露出時間の関係を確認し，オートガイダーとしての性能を調べてから範囲を定めることになる．

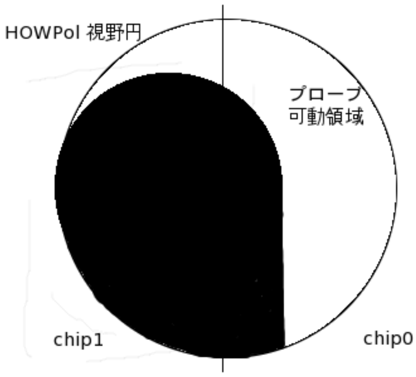


図 2.2: ガイド星選定可能領域概略図（黒塗りが禁止領域）

2.6 駆動系制御

ステージ制御は，モータドライバ CD-773 用の Windows ドライバに実装されているコマンド関数を用いる．各軸に対するモータの加速・減速時定数と，ステージ駆動の立ち上がりにおける最小速度と駆動中の最大速度の設定は，平木氏が設定した値をそのまま引き継いだ．現在は加速・減速ともに 50 mm/1000 Hz，最小速度を 3000 Hz，最大速度を 50000 Hz に設定してある．

本論文の付録に，オートガイダーのプログラムコードを掲載しているので，そちらも参照されたい．実際に制御のために送信している命令には以下の関数がある．

1. 環境設定ツール関数
   書式：BOOL ALK_EnvironmentInfo_Tool(ALK_S_RESULT FAR ∗*psResult*);
   役割：環境設定ツールで設定した情報で環境設定を行う．ステージ制御する際，この関数を必ず一番最初に記述しなければならない．

2. デバイスオープン関数
   書式：BOOL AC05_BOpen(WORD *IfNo*, WORD *SlaveAddr*, WORD *Axis*, WORD *SlaveType*, DWORD FAR ∗*phDev*, AC05_S_RESULT FAR ∗*psResult*)；
   役割：スレーブアドレス，軸，スレーブタイプを指定してデバイスをオープンする．

3. デバイスクローズ関数
   書式：BOOL AC05_BClose(DWORD *hDev*, AC05_S_RESULT FAR ∗*psResult*);
   役割：指定されたデバイスをクローズする．終了処理時に記述している．

4. データセット関数
   書式：VOID AC05_SetData(DWORD *Data*, AC05_S_DATA FAR ∗*psData*); 役割：24 ビットデータを *psResult* 構造体に格納して次の一括書き込み関数に利用する．

5. DRIVE COMMAND 一括書き込み関数
   書式：BOOL AC05_IWDrive(DWORD *hDev*, WORD Cmd, AC05_S_DATA FAR $*psData$, AC05_S_RESULT FAR $*psResult$);
   役割：指定されたデバイスのポートにコマンドコードを，データポートにデータを一括書き込みする．コマンド *Cmd* には 00H（初期化），12H/13H（+/-方向にドライブ）を用いて命令を送り，実際にステージを動かすなどする．

6. Status Port *n* 読み出し関数（*n* には 1，2 などが入る）
   書式：BOOL AC05_BRStatus*n*(DWORD *hDev*, WORD FAR $*pStatus$, AC05_S_RESULT FAR $*psResult$);
   役割：指定されたデバイスの指定ステータスポートを読み出す．ステータス 1 ポートは各々の軸の MCC05v2 の現在の状態を読み出すポート，ステータス 2 ポートは各々の軸の入力信号の状態を読み出すポートである．

2.7 CCD 制御

BITRAN 冷却 CCD カメラの通信では，PC 側がコマンドを出力して，実際に CCD 側が応答したコマンドに対しては CCD 側が返答する形がとられている．コマンドは 1 バイトを基本として，必要な場合はパラメータが続く．応答として戻りデータがくる場合がある．

CCD 制御に使用している関数は以下の通りである．

1. CCD デバイスのオープン
   書式：CCDOpenDevice()

2. カメラのリセット
   書式：CCDControllerReset()

3. CCD の情報を得る
   書式：CCDGetInfomation()

4. 撮影開始
   書式：CCDStartExposure(*Time*)
   Time：露出時間

5. 露出状態の取得
   書式：CCDExposingState()
   戻り値：露出の残り時間

6. 各ピクセルデータの取得
   書式：CCDTransferImage
7. 画像データの取り込み終了の確認
   書式：CCDFinishExposure()

本プログラムでは各ピクセルのカウントを16bit（65536階調）でデータを出力するように設定してあるが，その後データ処理の高速化のために8bit階調に落としている．将来的には16bitのデータをなるべく高速に扱えるようにし，精密な測光も可能なオートガイダーすることを目指している．

2.8　かなた望遠鏡の制御

オートガイダー制御用PCから望遠鏡へは，ソケット通信を用いて命令の送信や情報の受信を行っている．これはかなた望遠鏡の制御プログラム（UNIXマシン上で起動）がソケット通信による命令を受け付ける仕様となっているためである．このソケット通信プログラムも前任者より引き継いでいる．かなた望遠鏡とオートガイダーPCの通信の流れは以下の様になっている．

1. socket関数を呼び出してソケットを作成する
2. connect関数を呼び出してサーバに接続し，コネクションを確立する．
3. recvもしくはsend関数を呼び出し，送受信を行う．
4. close関数を呼び出して通信を終了させる．

望遠鏡の制御は2001年に当時名古屋大学大学院理学研究科Z研修士課程2年の加藤大輔氏によって作成されたソフトウェアを基に，西村製作所がアップグレードしたものを用いて行っている．ソケット通信を介して，オートガイダー制御用PCからこの望遠鏡制御ソフトウェアにコマンドを送ることが可能である．

望遠鏡の現在の情報を得るには
”A num1 num2 ... ...”
とし，numには取得したい値に対応した任意の望遠鏡のコマンドを入力した命令を送る．コマンドは情報ごとに番号が割りあてられており，例えば観測天体の赤経，赤緯の値を送ってもらう場合は
”A 38 39”
と言った命令を送る．今回のオートガイダーにおいては観測天体の座標を知るため，ある

いはガイド星の導入のために観測天体の赤経赤緯，また赤経・赤緯・ナスミスローテータのオフセット値を得たり，オートガイダー CCD で露光開始のタイミングを得るために望遠鏡が観測天体へ移動中かどうかといった情報を得るために使用している．

オートガイダーで得た追尾誤差（新たな赤経赤緯のオフセット）を望遠鏡に送るには
"P ra_off dec_off azi_off alt_off ir_off""
（各コマンドはそれ赤経・赤緯・方位・高度・ローテータ角のオフセット値）という命令を送り望遠鏡のオフセット値をセットする．この命令では，赤経・赤緯のオフセットだけを与えたい場合でも，同時に方位，高度，ローテータのオフセットを与える必要があるために，あらかじめ"A"コマンドで方位，高度，ローテータ角のオフセットも得ている．

第3章　試験観測と評価

3.1　オートガイダーの焦点調整とHOWPolとの視野合わせ

オートガイダーの焦点合わせと，オートガイダー視野とHOWPol視野の中心合わせを行った．

オートガイダーはその内部で焦点調整ができる構造になっている．焦点調整を行わないと星像のカウント分布が軸対称でなくなり，うまく重心を求めることができないため，焦点調整を行う必要がある．ただし，この焦点調整を行うと，Y軸ステージ方向に視野が動く機構になっているため，視野合わせを行う前に焦点調整を行う必要がある．この調整はピックアップミラーのステージの調整ネジにて行った．

焦点調整終了後，オートガイダー視野とHOWPol視野の中心合わせ行った．ピックアップミラーが搭載されたステージをステージ原点からガイド星にどれだけの距離を移動させるかは，HOWPolの視野中心の赤道座標とガイド星の赤道座標との相対位置から，ステージ移動パルス量に変換することで求めることができる．

中心合わせは

1. HOWPolの視野中心に恒星を導入
2. オートガイダーステージをHOWPolの視野中心付近へ移動
3. オートガイダーの視野中心にその恒星が導入されるようにステージの位置を微調整

という手順で行った．

この軸出しによって，ステージ原点からHOWPol視野中心へステージを動かすためのパルス送信量は

X = 255700（= 233 arcsec = 20.9 mm），Y = 236194（= 215 arcsec = 19.3 mm）

と求まった．この軸の位置を相対的な原点とすれば，視野中心軸とガイド星の座標はそれぞれ入手できるので，ステージ原点からガイド星へのパルス送信量を計算することができる．

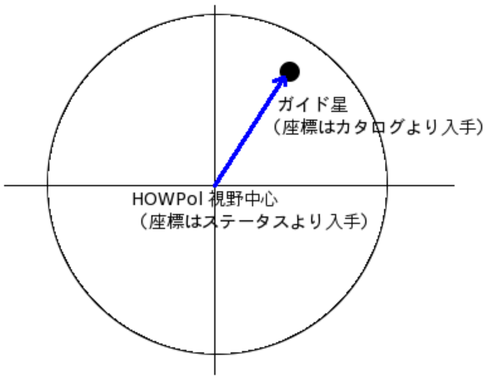


図 3.1: 軸出しによる相対的な原点

3.2　ガイド星自動検索の稼働状況

§2.3 のようにして，GSC1.2 から赤経，赤緯，等級だけを抜き出して一つのファイルにまとめたものに対して検索を行ったところ，プログラム実行開始から検索完了までに約 50 秒も掛かった．これでは時間が掛かり過ぎて，観測効率が悪い．そこで，検索速度向上のためにファイルを赤緯ごとに応じて分割した（表 3.1）．

表 3.1: 抽出した GSC の分割

赤緯 δ（°）	単位赤経（hour）	ファイル数
$+90 > \delta > +75$	24	1
$+75 \geq \delta > +60$	3	8
$+60 \geq \delta > +45$	3	8
$+45 \geq \delta > +30$	2	12
$+30 \geq \delta > +15$	2	12
$+15 \geq \delta > 0$	1	24
$0 \geq \delta > -15$	1	24
$-15 \geq \delta > -30$	2	12
$-30 \geq \delta > -45$	2	12

この結果，検索完了までの時間は約 1 秒にまで短縮された．検索結果は別のファイルに出力（図 3.2）し，ガイド星の決定にはこの出力ファイルを利用する．

検索結果が正しいかどうかは，GSC1.2 に対して簡便な条件での検索を行うことができる VizieR Service （ http://vizier.u-strasbg.fr/viz-bin/VizieR ）を利用して確認した．ラ

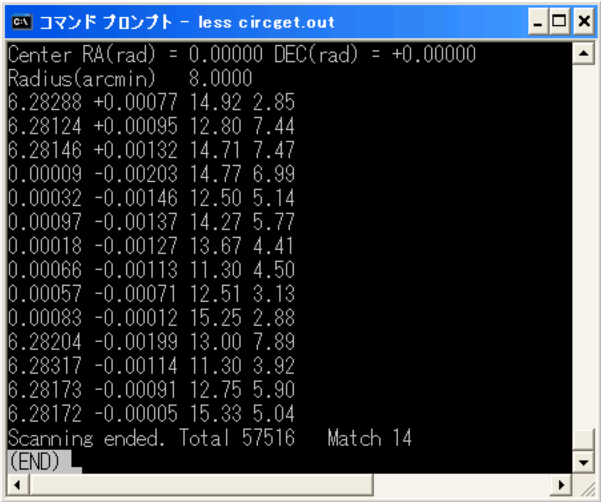


図 3.2: 検索結果出力ファイル（左から赤経（rad），赤緯（rad），等級，中心からの距離（arcmin））

ンダムに選んだ 10 個の座標において，このプログラムと VizieR を用いて検索を行い，結果を確認したところ 10 回全て一致した．これより抽出や分割は設計通りに行われていると判断した．

3.3 ガイド星選定時のステージ駆動

オートガイダーソフトウェア起動時には，オートガイダーステージの初期化として，各ステージを原点に移動する作業を行っている．しかし，平木氏のソフトでは θ 軸のリミットセンサの扱いに不具合があり，原点調整が行えないことが判明した．今回は θ 軸は使用せずに X，Y 軸だけ原点調整を行うことにした．原点調整にかかる時間は数秒未満と十分短く，安定してステージの初期化を行えるようになった．

テスト時，ガイド星選定アルゴリズムは十分に検討されていなかったので，望遠鏡が向いている座標から半径 3 分円から 8 分円の間で一番明るい天体を持ってくる，という簡単な条件でテストを行った．その結果，条件通りのガイド星をピックアップすること，ステージを移動させることはできたが，ステージの到達位置は正しくは無かった．この不具合の原因は現在調査中である．相対座標は正しく得られているため，相対座標からステージ移動パルス量への変換ミスなどが考えられる．今後，このバグを取り除いた上で再度テストを行う予定である．

3.4　CCD制御

CCD デバイスをオープン，初期化する際，セットした制御用コマンドがうまく動かない問題が発生した．これは Visual C++で用いられている CString の定義が，配布された CCD 制御用ライブラリにおける開発環境と，今回開発している環境とで異なる定義であることが原因であった．そのため，CString で書かれている部分を char 型に書き換えることによってこの問題は解決した．CString 型では文字列の長さに応じてメモリ量が自動で割り当てられるが，char 型では手動でメモリ量を割り当てなければならないため，メモリ割り当て不足の無いように適切に配列の大きさなどを割り当てる．

また CCD データの画像化も行った．CCD によって得られたデータからは，本来は輝度分布だけ計算できれば良いが，画像として確認できるようにしないと，正しく計算が行われ，オートガイドしているかどうか確認しにくい．そこで，本プログラムでは CCD データを fits 形式として画像化するようにした．fits 形式は天文学分野で広く用いられている標準的な画像フォーマットであり，fits 画像ビュアーも充実しているため，簡便に扱うことができる．実際には，CFITSIO というサブルーチンライブラリを用いて CCD データを fits 画像にしている．

fits データの表示には，図 3.3 のように DS9 というビュアーを利用している．

3.5　オートガイド機能

オートガイダーの手順は，CCD に露光し，ピクセルごとのデータをメモリに格納後，最初の重心を求め，再び露光して重心を求めて最初の重心とのズレを算出し，それを修正する方向に望遠鏡を動かすというものである．2 回目の露光から望遠鏡に情報を送信するところまでが繰り返し部分となり，繰り返しごとに最初の重心との比較を行う．

オートガイドプログラムが一通り組み上がったところで，手動で適当な星を導入して，オートガイド機能のルーチンを実行させた．

実行後，繰り返し部分に入ったところでランタイムエラーを起こしプログラムが終了してしまうことが判明した．はっきりとした原因はまだ判明していないが，望遠鏡にオフセットをセットする部分は，正しく動作していたので，CCD データの取り扱い部分に依然バグがあると思われる．これについても，今後早急に解決を目指す．

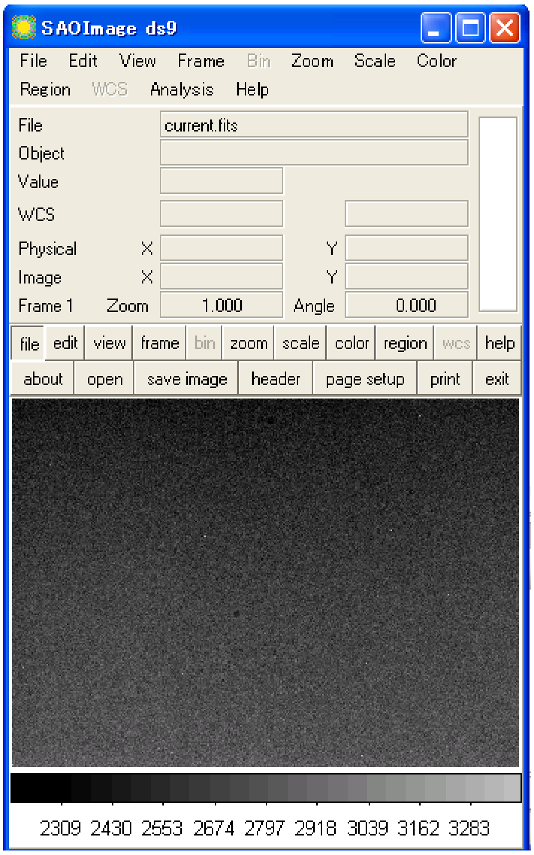


図 3.3: DS9 で表示されたオートガイダー CCD データ

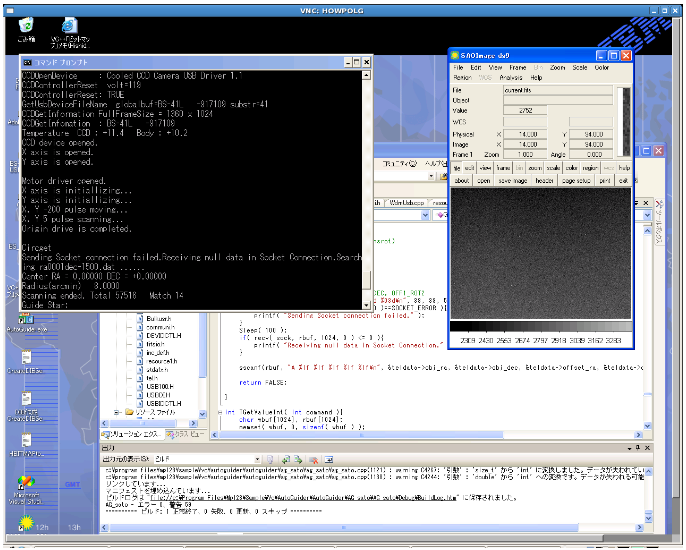


図 3.4: プログラム実行中の画面

第4章　まとめと今後の課題

本研究では，HOWPol に組み込まれているオートガイダーを全自動で制御することを目指した．これまでに完了した項目としては，制御プログラムの更新と，ガイド星自動選定プログラムが挙げられる．

制御プログラムの更新については，

1. オートガイダーステージの初期化・自動制御
2. オートガイダー CCD の初期化・露出制御・取得データ制御・取得データの可視化
3. ソケット通信を介した望遠鏡へ情報の送受信

を自動で行えるようにした．オートガイダー CCD データは，CFITSIO を用いて fits 画像形式に変換し，DS9 で表示するようにした．

また，ガイド星自動選定については，まず，恒星カタログの整理を行った．Guide Star Catalog1.2 からデータを抽出し，一つのファイルにまとめた後，高速に検索するため，このファイルを 113 個に分割し，結果として約 1 秒で情報を取得できることができるようにした．

また，オートガイダーを使用する場合，HOWPol 視野の一部がピックアッププローブの影によって遮蔽されてしまうため，プローブの可動領域を考慮してガイド星を選定する必要があり，プローブの大きさと望遠鏡の光学系から HOWPol の視野を遮蔽する影の大きさを計算し，ガイド星を選定する領域を制限した．

今後，オートガイダーを完成させるには，次に述べる項目を完成させなくてはならない．

1. オートガイダー視野中心にガイド星が導入されるようにステージを正しく移動させる．
2. オートガイダーの CCD の露出を開始し，取得したデータからガイド星像の輝度重心を求める．
3. 再び露出を行い，最初の重心からのずれを求め，望遠鏡にずれを修正する方向にオフセット値を送る．

4. 最初に求めた重心の位置は保持しておき，再露出から望遠鏡へのずれ修正までのルーチンを繰り返すようにする．

現在，繰り返し部分に入ったところでランタイムエラーが出るバグがある．これはCCDのデータをメモリに転送した後に出ているエラーと言うところまでは分かったが，未解決のままである．

以上の課題をなるべく早く克服して，オートガイダーを実用化させたい．

付 録A

A.1 ソースコード

```
// AG_sato.cpp : コンソール アプリケーションのエントリ ポイントを定義します。
//
#include "stdafx.h"

#include <stdlib.h>
#include <math.h>
#include <string.h>
#include <string>
#include <conio.h>
#include "tel.h"
#include "inc_def.h"

//#include <afx.h>

#include "Ac05A.h"
#include "AlUsbA.h"
#include "AcIoA.h"
#include "Bulkusr.h"
#include "communi.h"

#include <windows.h>

#include "fitsio.h"


char str[300];
long cur_pulse_x, cur_pulse_y;

long center_pulse_x = 255700;
long center_pulse_y = 236194;

float orig_nsrot = -63.8;
float thetaorg = 15.0;

//#define DIB_HEADER_MARKER   ((WORD) ('M' << 8) | 'B')   // "BM"
//#define PALVERSION   0x300          // DIB constants
//#define MAXPALCOLORS 256
#define NX 340
#define NY 256
#define nx 50
#define ny 50
//#define PI 3.1416
//#define TELFLAG 0
#define C_low 0
#define C_high 65535

static DWORD                                    hDev1;                                   // デバイスハンドル変数
static DWORD                                    hDev2;                                   // デバイスハンドル変数

static AC05_S_DATA                              Ac05Data;                                // データエリア
static AC05_S_RESULT                    Ac05Result;                              // RESULT 格納エリア
static ACIO_S_RESULT                    AcIoResult;                              // RESULT 格納エリア
static WORD                                             Cmd;                                             // コマンド設定変数
static WORD                                             Data;                                            // データ設定変数

static ALK_S_RESULT                             AlkResult;                               // RESULT 格納エリア

static WORD                                             Status1_x;
static WORD                                             Status1_y;
static WORD                                             Status2_x;
static WORD                                             Status2_y;

char *mid( char *buf, int ns, int nn ){
```

```
        int i;
        for( i=0; i<=(nn-1); i++ ){
                str[i] = buf[ns+i-1];
        }
        str[nn] = '¥0';
        return( str );
}

//
// Open X axis port
//
void XAxiOpen(void){

        if( AC05_BOpen( AC05_USB, 1, AC05_X, AC05_SLAVE_CD773, &hDev1, &Ac05Result ) == 0 ){
                printf("Open Error X axis.¥n");
                Sleep(1000);
                exit(-1);
        }
        Cmd = 0xF9;
        AC05_BWDriveCommand( hDev1, &Cmd, &Ac05Result );
        printf("X axis is opened.¥n");
}

//
// Open Y axis port
//
void YAxiOpen(void){
        if(AC05_BOpen( AC05_USB, 1, AC05_Y, AC05_SLAVE_CD773, &hDev2, &Ac05Result ) == 0){
                printf("Open Error Y axis.¥n");
                Sleep(1000);
                exit(-1);
        }
        Cmd = 0xF9;
        AC05_BWDriveCommand( hDev2, &Cmd, &Ac05Result );
        printf("Y axis is opened.¥n¥n");
}

//
// Close X axis port
//
void XAxiClose(void){
        AC05_BClose( hDev1, &Ac05Result );
        printf("X axis is closed.¥n");
}

//
// Close Y axis port
//
void YAxiClose(void){
        AC05_BClose( hDev1, &Ac05Result );
        printf("Y axis is closed.¥n");
}

long PulseCounter_x(long input_pulse_x){
        cur_pulse_x += input_pulse_x;
        return(cur_pulse_x);
}

long PulseCounter_y(long input_pulse_y){
        cur_pulse_y += input_pulse_y;
        return(cur_pulse_y);
}


void OriginDrive(){

        AC05_BWaitDriveCommand( hDev1, 0x00, &Ac05Result );
        AC05_BWaitDriveCommand( hDev2, 0x00, &Ac05Result );
        //AC05_BWaitDriveCommand( hDev3, 0x00, &Ac05Result );

        Cmd = 0x12;
        AC05_BWDriveCommand( hDev1, &Cmd, &Ac05Result );
        printf("X axis is initiallizing...¥n");
        //ReadyWait( &Ac05Result, &StopCode );
        Cmd = 0x12;
        AC05_BWDriveCommand( hDev2, &Cmd, &Ac05Result );
        printf("Y axis is initiallizing...¥n");
        //ReadyWait2( &Ac05Result, &StopCode );
        WaitLimit();

        printf("X, Y -200 pulse moving...¥n");

        AC05_SetData( (DWORD)-200, &Ac05Data );
        AC05_IWDrive( hDev1, 0x14, &Ac05Data, &Ac05Result );
```

```
        AC05_IWDrive( hDev2, 0x14, &Ac05Data, &Ac05Result );
        //ReadyWait( &Ac05Result, &StopCode );
        //ReadyWait2( &Ac05Result, &StopCode );
        WaitNotBusy();

        printf("X, Y 5 pulse scanning...¥n");

        AC05_BRStatus1( hDev1, &Status1_x, &Ac05Result );
        AC05_BRStatus1( hDev2, &Status1_y, &Ac05Result );

        do{
                AC05_SetData( 5, &Ac05Data );
                if((Status1_x & 0x08) == 0x00){
                        AC05_IWDrive( hDev1, 0x14, &Ac05Data, &Ac05Result );
                        WaitNotBusy();
                }
                else printf("X axis  completed.¥n");
                //ReadyWait( &Ac05Result, &StopCode );
                //ReadyWait2( &Ac05Result, &StopCode );
                AC05_BRStatus2( hDev1, &Status2_x, &Ac05Result );
                // NowAddressDisplay( &Ac05Result );
        }while( (Status2_x & 0x08) == 0x00);


        do{
                AC05_SetData( 5, &Ac05Data );
                if((Status1_y & 0x08) == 0x00){
                        AC05_IWDrive( hDev2, 0x14, &Ac05Data, &Ac05Result );
                        WaitNotBusy();
                }
                else printf("Y axis  completed.¥n");
                //ReadyWait( &Ac05Result, &StopCode );
                //ReadyWait2( &Ac05Result, &StopCode );
                AC05_BRStatus2( hDev2, &Status2_y, &Ac05Result );
                // NowAddressDisplay( &Ac05Result );
        }while( (Status2_y & 0x08) == 0x00 );


        // NowAddressDisplay( &Ac05Result );

        // counter reset
        cur_pulse_x = 0;
        cur_pulse_y = 0;
        printf("Origin drive is completed.¥n¥n");

}

void WaitNotBusy(){
        //char flagx, flagy;

        flagx = 0;
        flagy = 0;

        while( flagx == 0 || flagy == 0){
                if( flagx == 0){
                        AC05_BRStatus1( hDev1, &Status1_x, &Ac05Result );
                        // NowAddressDisplay( &Ac05Result );
                        if( (Status1_x & 0x01) == 0x00 ){
                                flagx = 1;
                        }
                }
                if( flagy == 0){
                        AC05_BRStatus1( hDev2, &Status1_y, &Ac05Result );
                        // NowAddressDisplay( &Ac05Result );
                        if( (Status1_y & 0x01) == 0x00 ){
                                flagy = 1;
                        }
                }
        }
}

void WaitLimit(){
        //char flagx, flagy;

        flagx = 0;
        flagy = 0;

        while( flagx == 0 || flagy == 0){
                if( flagx == 0){
                        AC05_BRStatus2( hDev1, &Status2_x, &Ac05Result );
                        // NowAddressDisplay( &Ac05Result );
                        if( Status2_x & 0x08 ){
                                flagx = 1;
                        }
```

```
		}
		if( flagy == 0){
			AC05_BRStatus2( hDev2, &Status2_y, &Ac05Result );
			// NowAddressDisplay( &Ac05Result );
			if( Status2_y & 0x08 ){
				flagy =1;
			}
		}
	}
}


void RateSet(void){
	/*** RATE SET ***/                                              //X 軸の RATESET
	//   DRATE : 10ms/1000Hz       URATE : 10ms/1000Hz
	AC05_BWaitDriveCommand( hDev1, 0, &Ac05Result );
	Data = 0x11;
	AC05_BWDriveData2( hDev1, &Data, &Ac05Result );
	Data = 0x11;
	AC05_BWDriveData3( hDev1, &Data, &Ac05Result );
	Cmd = 0x06;
	AC05_BWDriveCommand( hDev1, &Cmd, &Ac05Result );

	/*** LSPD SET ***/
	//   LSPD : 1000Hz
	AC05_BWaitDriveCommand( hDev1, 0, &Ac05Result );
	AC05_SetData( 3000, &Ac05Data );
	AC05_IWDrive( hDev1, 0x07, &Ac05Data, &Ac05Result );

	/*** HSPD SET ***/
	//   HSPD : 5000Hz
	AC05_BWaitDriveCommand( hDev1, 0, &Ac05Result );
	AC05_SetData( 50000, &Ac05Data );
	AC05_IWDrive( hDev1, 0x08, &Ac05Data, &Ac05Result );

	/*** RATE SET ***/                                              //Y 軸の RATESET
	//   DRATE : 10ms/1000Hz       URATE : 10ms/1000Hz
	AC05_BWaitDriveCommand( hDev2, 0, &Ac05Result );
	Data = 0x11;
	AC05_BWDriveData2( hDev2, &Data, &Ac05Result );
	Data = 0x11;
	AC05_BWDriveData3( hDev2, &Data, &Ac05Result );
	Cmd = 0x06;
	AC05_BWDriveCommand( hDev2, &Cmd, &Ac05Result );

	/*** LSPD SET ***/
	//   LSPD : 1000Hz
	AC05_BWaitDriveCommand( hDev2, 0, &Ac05Result );
	AC05_SetData( 3000, &Ac05Data );
	AC05_IWDrive( hDev2, 0x07, &Ac05Data, &Ac05Result );

	/*** HSPD SET ***/
	//   HSPD : 5000Hz
	AC05_BWaitDriveCommand( hDev2, 0, &Ac05Result );
	AC05_SetData( 50000, &Ac05Data );
	AC05_IWDrive( hDev2, 0x08, &Ac05Data, &Ac05Result );

}

void Jushin(unsigned int pix[][NX], double *x_grav, double *y_grav, double *SkyLev){
	int i, j, x, y;
	double sigma_x = 0.0, sigma_y = 0.0, count = 0.0;
	double Sky = 0.0;
	int n_Sky = 0;
	x = NX;
	y = NY;


	for( j = 0; j <= 10; j++)
	{
		for( i = 0; i <= 10; i++)
		{
			Sky += pix[j][i];
			n_Sky++;
		}
	}

	for( j = 0; j <= 10; j++)
	{
		for( i = x-11; i <= x-1; i++)
		{
			Sky += pix[j][i];
			n_Sky++;
		}
```

```
        }

        for( j = y-11; j <= y-1; j++)
        {
                for( i = 0; i <= 5; i++)
                {
                        Sky += pix[j][i];
                        n_Sky++;
                }
        }

        for( j = y-11; j <= y-1; j++)
        {
                for( i = x-11; i <= x-1; i++)
                {
                        Sky += pix[j][i];
                        n_Sky++;
                }
        }

        *SkyLev = Sky / n_Sky;

        for( j = 0; j < y; j++)
        {
                for( i = 0; i < x; i++)
                {
                        sigma_x += i * (pix[j][i] - *SkyLev);
                        sigma_y += j * (pix[j][i] - *SkyLev);
                        count += ( pix[j][i] - *SkyLev );
                }
        }

        *x_grav = sigma_x / count;
        *y_grav = sigma_y / count;

        if( *x_grav < 0.0 )  *x_grav = x / 2.0;
        if( *x_grav > x ) *x_grav = x / 2.0;
        if( *y_grav < 0.0 )  *y_grav = y / 2.0;
        if( *y_grav > y ) *y_grav = y / 2.0;
}

void Maximam(unsigned int pix[][NX], int &M_x, int &M_y, int &M_count){
        int i, j;
        int Max = -100;

        for ( j = 0; j < NY; j++)
        {
                for ( i = 0; i < NX; i++)
                {
                        if(Max <= pix[i][j])
                        {
                                Max = pix[i][j];
                                M_x = i;
                                M_y = j;
                        }

                }
        }
        M_count = Max;

}

//
// Get current telescope state
//
void pos_now(double *center_ra, double *center_dec, double *nsrot){
        double RA_base, DEC_base;
        double offset_RA, offset_DEC, offset_NsROT;

        TELEPARAM *teldata;
        teldata = (TELEPARAM *)malloc(sizeof(TELEPARAM));
        memset((TELEPARAM *)teldata, 0.0, sizeof(TELEPARAM));

        // Initialize Socket
        //SockInit();
        //SockConnect();

        // get radec, offset_radec, nsrot
        // R_A_OBJ, DEC_OBJ, OFF1_R_A, OFF1_DEC, OFF1_ROT2
        GetTelValue(teldata);

        RA_base = teldata->obj_ra;
        DEC_base = teldata->obj_dec;
        offset_RA = teldata->offset_ra;
```

```
        offset_DEC = teldata->offset_dec;
        offset_NsROT = teldata->nsrot;

        *center_ra = RA_base + offset_RA;
        *center_dec = DEC_base + offset_DEC;
        *nsrot = offset_NsROT;

        free(teldata);

        //closesocket(sock);
        //sock=INVALID_SOCKET;
}

int circget(void){

        double Deg2Rad = 0.017453292; /* PI/180 */
        int i;

        FILE *fp1, *fp2;
        char buf[250], last_buf[250], fbuf[50];
        double a1, d1, a2, d2;
        //      double rahms, decdms, ras, decs;
        double ratel, dectel;
        //      int rah, ram;
        //      int decd, decm;
        double rarad, decrad;
        double rah_orig, decd_orig;
        double decsign;
        double nsrot;

        char f2[] = "circget.out"; // output file name
        double radius = 8.; // search radius

        double coscrit, sind1, cosd1;
        double cosd2, sind2, cosaa, cosc;

        int n=0, nm=0;

        int di;

        /*** center position ***/

        /*
           rah = rahms / 10000.0;
           ram = (rahms - rah*10000.0) / 100.0;
           ras = rahms - rah*10000.0 - ram*100.0;
           rah_orig = rah + ram/60.0 + ras/3600.0;
           rarad = rah_orig * 15.0 * Deg2Rad;
           a1 = rarad;
         */
        /*
           decd = decdms / 10000.0;
           decm = (decdms - decd*10000.0) / 100.0;
           decs = decdms - decd*10000.0 - decm*100.0;
           decd_orig = decsign * ( decd + decm/60.0 + decs/3600.0);
           decrad = decd_orig * Deg2Rad;
           d1 = decrad;
         */

        pos_now(&ratel, &dectel, &nsrot);

        rarad = ((ratel / 3600) * 15) * Deg2Rad;
        a1 = rarad;
        rah_orig = (rarad / 15.0) / Deg2Rad;


        decrad = (dectel / 3600) * Deg2Rad;
        d1 = decrad;
        decd_orig = decrad / Deg2Rad;

        if( decrad < 0.0 ){
                decsign = -1.0;
                decrad *= -1.0;
        }
        else
                decsign = +1.0;


        if(75 < decd_orig && decd_orig <=90){
                strcpy( fbuf, "ra0024dec7590.dat" );
                if( ( fp1 = fopen( fbuf, "rt" ) ) == NULL ){
                        printf( "Can't open gsc file¥n" );
                        exit( -1 );
                }
```

```
}
else if(60 < decd_orig && decd_orig <= 75){
        di = 3;
        for(i=0; i<24; i+=di){
                if(i <= rah_orig && rah_orig < i+di){
                        sprintf(fbuf, "ra%02d%02ddec6075.dat", i, i+di);
                        if( ( fp1 = fopen( fbuf, "rt" ) ) == NULL ){
                                printf( "Can't open gsc file.¥n" );
                                exit( -1 );
                        }
                }
        }
}
else if(45 < decd_orig && decd_orig <= 60){
        di = 3;
        for(i=0; i<24; i+=di){
                if(i <= rah_orig && rah_orig < i+di){
                        sprintf(fbuf, "ra%02d%02ddec4560.dat", i, i+di);
                        if( ( fp1 = fopen( fbuf, "rt" ) ) == NULL ){
                                printf( "Can't open gsc file.¥n" );
                                exit( -1 );
                        }
                }
        }
}
else if(30 < decd_orig && decd_orig <= 45){
        di = 2;
        for(i=0; i<24; i+=di){
                if(i <= rah_orig && rah_orig < i+di){
                        sprintf(fbuf, "ra%02d%02ddec3045.dat", i, i+di);
                        if( ( fp1 = fopen( fbuf, "rt" ) ) == NULL ){
                                printf( "Can't open gsc file.¥n" );
                                exit( -1 );
                        }
                }
        }
}
else if(15 < decd_orig && decd_orig <= 30){
        di = 2;
        for(i=0; i<24; i+=di){
                if(i <= rah_orig && rah_orig < i+di){
                        sprintf(fbuf, "ra%02d%02ddec1530.dat", i, i+di);
                        if( ( fp1 = fopen( fbuf, "rt" ) ) == NULL ){
                                printf( "Can't open gsc file.¥n" );
                                exit( -1 );
                        }
                }
        }
}
else if(0 < decd_orig && decd_orig <= 15){
        di = 1;
        for(i=0; i<24; i+=di){
                if(i <= rah_orig && rah_orig < i+di){
                        sprintf(fbuf, "ra%02d%02ddec0015.dat", i, i+di);
                        if( ( fp1 = fopen( fbuf, "rt" ) ) == NULL ){
                                printf( "Can't open gsc file.¥n" );
                                exit(-1);
                        }
                }
        }
}
else if(-15 < decd_orig && decd_orig <= 0){
        di = 1;
        for(i=0; i<24; i+=di){
                if(i <= rah_orig && rah_orig < i+di){
                        sprintf(fbuf, "ra%02d%02ddec-1500.dat", i, i+di);
                        if( ( fp1 = fopen( fbuf, "rt" ) ) == NULL ){
                                printf( "Can't open gsc file.¥n" );
                                exit( -1 );
                        }
                }
        }
}
else if(-15 < decd_orig && decd_orig <= 0){
        di = 1;
        for(i=0; i<24; i+=di){
                if(i <= rah_orig && rah_orig < i+di){
                        sprintf(fbuf, "ra%02d%02ddec-1500.dat", i, i+di);
                        if( ( fp1 = fopen( fbuf, "rt" ) ) == NULL ){
                                printf( "Can't open gsc file.¥n" );
                                exit( -1 );
                        }
                }
        }
```

```
}
else if(-30 < decd_orig && decd_orig <= -15){
        di = 2;
        for(i=0; i<24; i+=di){
                if(i <= rah_orig && rah_orig < i+di){
                        sprintf(fbuf, "ra%02d%02ddec-30-15.dat", i, i+di);
                        if( ( fp1 = fopen( fbuf, "rt" ) ) == NULL ){
                                printf( "Can't open gsc file.¥n" );
                                exit( -1 );
                        }
                }
        }
}
else if(-45 < decd_orig && decd_orig <= -30){
        di = 2;
        for(i=0; i<24; i+=di){
                if(i <= rah_orig && rah_orig < i+di){
                        sprintf(fbuf, "ra%02d%02ddec-45-30dat", i, i+di);
                        if( ( fp1 = fopen( fbuf, "rt" ) ) == NULL ){
                                printf( "Can't open gsc file.¥n" );
                                exit( -1 );
                        }
                }
        }
}

printf("Searching %s ......¥n", fbuf);
/*** open output file ***/
if( ( fp2 = fopen( f2, "wt" ) ) == NULL ){
        printf( "Open error '%s'¥n", f2 );
        exit( -1 );
}

printf( "Center RA = %07.5f DEC = %+07.5f¥n", rarad, decrad );
fprintf( fp2, "Center RA(rad) = %07.5f DEC(rad) = %+07.5f¥n", rarad, decrad );

/*** circle radius ***/

printf( "Radius(arcmin)  %7.4f¥n", radius );
fprintf( fp2, "Radius(arcmin)  %7.4f¥n", radius );

radius = (radius/60.0) * Deg2Rad;
coscrit = cos( radius );
sind1 = sin( d1 );
cosd1 = cos( d1 );


/*** scan catalogue file ***/

while( feof( fp1 ) == 0 ){
        fgets( buf, 250, fp1 );
        if( strcmp( buf, last_buf ) == 0 )
                continue;

        n++;

        a2 = atof( mid( buf, 1, 7 ) );
        d2 = atof( mid( buf, 9, 8 ) );

        if( a2 > 6.28319 || a2 < 0.0 || d2 > 1.57080 || d2 < -1.57080 ){
                printf( "Data error in '%s (a2=%e d2=%e)'¥n", fbuf, a2, d2 );
                printf( "%s", buf );
                fclose( fp1 );
                fclose( fp2 );
                exit( -1 );
        }

        cosd2 = cos( d2 );
        sind2 = sin( d2 );
        cosaa = cos( a1 - a2 );

        cosc = sind1*sind2 + cosd1*cosd2*cosaa;

        /*** compare ***/
        if( cosc > coscrit ){
                radius = acos( cosc );
                radius = (radius / Deg2Rad) * 60;
                nm++;
                fprintf( fp2, "%s %4.2f¥n", mid( buf, 1, ( strlen( buf ) -1 ) ), radius );
        }

        strcpy( last_buf, buf );
}
```

```
        printf( "Scanning ended. Total %d   Match %d¥n", n, nm );
        fprintf( fp2, "Scanning ended. Total %d   Match %d¥n", n, nm );

        fclose( fp2 );
        fclose( fp1 );

        return( nm );
}


//
// Determine guide star
//
int detgstar(void){

        char *circgetout = "circget.out";

        // double arcsec2pix = 12.5;
        double pix2pulse = 10000 / 310;
        double deg2rad = 0.017453292; /* PI/180 */

        char buf[200], last_buf[200];
        char buf1[100], buf2[100], buf3[100], buf4[100], buf5[50], buf6[50], buf7[50];
        char ratmpstr[20] = "0.00000", dectmpstr[20] = "0.00000", magtmpstr[20] = "00.00", radiustmpstr[20] = "0.00";
        char center_rastr[20] = "0.00000", center_decstr[20] = "0.00000";
        double ratmp, dectmp, magtmp, radiustmp;
        double ramax, decmax, magmax = 16.1, radiusmax = 0;
        double center_ra, center_dec;
        double ra_guide, dec_guide, mag_guide, radius_guide;
        double dra, ddec;
        double nsrot;
        double ratel, dectel;
        double rotate_dra, rotate_ddec;
        double arcsec_dra, arcsec_ddec;
        double orig2gstar_pulse_x, orig2gstar_pulse_y;
        double dtheta;
        double SF; SF = 0.072 * 4; // (arcsec/pix)

        FILE *fp;

        if((fp = fopen(circgetout, "rt")) == NULL){
                printf("Cannot open circget output file '%s'¥n", circgetout);
        }
        else{
                while(feof(fp) == 0){
                        fgets(buf, 200, fp);
                        if(strcmp(buf, last_buf) == 0)
                                continue;
                        sscanf(buf, "%s %s %s %s %s %s %s", buf1, buf2, buf3, buf4, buf5, buf6, buf7);
                        if(strcmp(buf1, "Center") != 0 && strcmp(buf1, "Radius(arcmin)" ) != 0 && strcmp(buf1, "Scanning") != 0){
                                strcpy(ratmpstr,  buf1); ratmp = atof(ratmpstr);
                                strcpy(dectmpstr, buf2); dectmp = atof(dectmpstr);
                                strcpy(magtmpstr, buf3); magtmp = atof(magtmpstr);
                                strcpy(radiustmpstr, buf4); radiustmp = atof(radiustmpstr);

                                if(magmax > magtmp && radiustmp > 3.00){
                                        ramax = ratmp;
                                        decmax = dectmp;
                                        magmax = magtmp;
                                        radiusmax = radiustmp;
                                }
                        }
                        if(strcmp(buf1, "Center") == 0){
                                strcpy(center_rastr, buf4); center_ra = atof(center_rastr);
                                strcpy(center_decstr, buf7); center_dec = atof(center_decstr);
                        }
                }
                ra_guide = ramax;
                dec_guide = decmax;
                mag_guide = magmax;
                radius_guide = radiusmax;

                if(magmax == 16.1 && radiusmax == 0){
                        printf("Please select a guide star by yourself.¥n");
                }
                else{
                        printf("Guide Star:¥nRA(rad):%07.5f DEC(rad):%+08.5f mag:%05.2f radius(arcmin):%04.2f¥n", ra_guide, dec_guide, mag_guide, radius_guide);
                }

        }

        // Moving stage to guide star
        dra = ra_guide - center_ra;
        ddec = dec_guide - center_dec;
```

```
        pos_now(&ratel, &dectel, &nsrot);
        dtheta = (nsrot - orig_nsrot) * deg2rad;

        rotate_dra = dra * cos(dtheta) - ddec * sin(dtheta);
        rotate_ddec = dra * sin(dtheta) + ddec * cos(dtheta);

        arcsec_dra = (rotate_dra/deg2rad) * 3600;
        arcsec_ddec = (rotate_ddec/deg2rad) * 3600;

        orig2gstar_pulse_x = center_pulse_x + ((arcsec_dra / SF) * pix2pulse);
        orig2gstar_pulse_y = center_pulse_y + ((arcsec_ddec / SF) * pix2pulse);

        printf("Send pulse to X stage: %d¥n", (DWORD)orig2gstar_pulse_x);
        printf("Send pulse to Y state: %d¥n", (DWORD)orig2gstar_pulse_y);

        /*
           AC05_BWaitDriveCommand( hDev1, 0, &Ac05Result );
           AC05_SetData( (DWORD)-orig2gstar_pulse_x, &Ac05Data );
           AC05_IWDrive( hDev1, 0x14, &Ac05Data, &Ac05Result );
           WaitNotBusy();
           PulseCounter_x((long)-orig2gstar_pulse_x);

        //ReadyWait( &Ac05Result, &StopCode );
        //StopCodeDisplay( StopCode );

        AC05_BWaitDriveCommand( hDev2, 0, &Ac05Result );
        AC05_SetData( (DWORD)-orig2gstar_pulse_y, &Ac05Data );
        AC05_IWDrive( hDev2, 0x14, &Ac05Data, &Ac05Result );
        WaitNotBusy();
        PulseCounter_y((long)-orig2gstar_pulse_y);
        //ReadyWait2( &Ac05Result, &StopCode );
        //StopCodeDisplay( StopCode );
        printf("X,Y stage is moved to guide star.¥n");
         */
        return(0);

}


//
// FITS write image using cfitsio
//

int writefitsimage( unsigned int pixdata[][NX] ){

        fitsfile *fp;
        char *outfile = "!current.fits"; // '!' allows overwrite if it already exists.
        int status;
        long fitsnaxes[2];

        fitsnaxes[0] = NX;
        fitsnaxes[1] = NY;

        if( fits_create_file( &fp, outfile, &status ) != 0 ){
                printf( "Cannot create file '%s'. status=%d¥n",¥
                                outfile, status );
                return( status );
        }

        if( fits_create_img( fp, USHORT_IMG, 2, fitsnaxes, &status ) != 0 ){
                printf( "Cannot create image header. status=%d¥n",¥
                                status );
                return( status );
        }

        if( fits_write_img( fp, TUINT, 1, fitsnaxes[0]*fitsnaxes[1], ¥
                                pixdata, &status ) != 0 ){
                printf( "Cannot write image data. status=%d¥n",¥
                                status );
                return( status );
        }

        if( fits_close_file( fp, &status ) != 0 ){
                printf( "Cannot close file '%s' normall. status=%d¥n",¥
                                outfile, status );
                return( status );
        }

        return( 0 );
}

void AutoGuide(void){
```

```
CSize size;
HGLOBAL hCCD = NULL;
long Ti;
double exp_time = 1.0;
int i, j;
double nsrot;
double ratel, dectel;
double dtheta;
double SF; SF = 0.072 * 4;
double deg2rad = 0.017453292; /* PI/180 */

//      unsigned int pixel[NX][NY];
unsigned int pixdata[NY][NX];

// Exposure Time
Ti = exp_time * 1000.;

// CCD Environmental Setting
BYTE data1[] = {0x20, 0x10, 0, 0, 0};
OutPort(data1, sizeof(data1));

// Binning Setting (16bit, 4x4 binning, all pixel)
size = CCDSizeExposure(0, 3, 0);
//size.cx = NX;
//size.cy = NY;

// check whether telescope is tracking or not

while( TGetValueInt( 138 ) == 0 ){
        Sleep( 100 );
}

// exposure starting
printf( "Exposure started. Exp_time %ld msec¥n", Ti );
CCDStartExposure( Ti );
while( CCDExposingState() != 0 ){
        Sleep( 100 );
}
printf( "Exposure finished¥n" );

// Exposure Setting
//BYTE data3[] = {0x22, LOBYTE(Ti), HIBYTE(Ti), 0};
//OutPort(data3, sizeof(data3));

// Beginning Exposure
//BYTE data4[] = {0x23, 0};

//Sleep((DWORD)Ti*100);


// Transfer Image Data
hCCD=::GlobalAlloc(GMEM_MOVEABLE | GMEM_ZEROINIT, size.cx * size.cy * 2);
CCDTransferImage(0, hCCD);

// Finish Exposure();
CCDFinishExposure();

LPBYTE pCCD = (LPBYTE) ::GlobalLock(hCCD);


for ( j = NY-1; j >= 0; j--)
{
        for ( i = 0; i < NX; i++)
        {
                ULONG data = MAKEWORD(pCCD[0], pCCD[1]);                                //A/D 変換 16bit の時
                //pixdata[i][j] = data;
                pixdata[j][i] = data;
                pCCD += 2;
        }
}
::GlobalUnlock(hCCD);
::GlobalFree(hCCD);

printf( "Pixdata  %d %d %d¥n", pixdata[50][71], pixdata[50][72], pixdata[50][73] );
printf( "Pixdata  %d %d %d¥n", pixdata[51][71], pixdata[51][72], pixdata[51][73] );
printf( "Pixdata  %d %d %d¥n", pixdata[52][71], pixdata[52][72], pixdata[52][73] );

printf( "Write fits file... status %d¥n", writefitsimage( pixdata ) );


// convert to 8bit data
unsigned int pixel[NY][NX];

for( j = 0; j <NY; j++){
```

```
        for( i = 0; i < NX; i++){
                pixel[j][i] = (pixdata[j][i] - C_low) * 256 / (C_high - C_low);
                if(pixel[j][i] < 0)
                        pixel[j][i] = 0;
                else if(pixel[j][i] > 255)
                        pixel[j][i] = 255;
        }
}


double GX, GY, LOS;
double guid_GX, guid_GY;
double dGX, dGY;
double rotate_dGX, rotate_dGY;
double arcsec_dra, arcsec_ddec;

// remove skylevel(LOS) and get center of gravity(GX,GY)
Jushin(pixel, &guid_GX, &guid_GY, &LOS);
printf("First center of gravity is getted¥n");

/*
   while(1){
// check whether telescope is tracking or not
while( TGetValueInt( 138 ) == 0 ){
Sleep( 100 );
}

// exposure starting
printf( "Exposure started. Exp_time %ld msec¥n", Ti );
CCDStartExposure( Ti );
while( CCDExposingState() != 0 ){
Sleep( 100 );
}
printf( "Exposure finished¥n" );


// Transfer Image Data
hCCD=::GlobalAlloc(GMEM_MOVEABLE | GMEM_ZEROINIT, size.cx * size.cy * 2);
CCDTransferImage(0, hCCD);

// Finish Exposure();
CCDFinishExposure();

LPBYTE pCCD = (LPBYTE) ::GlobalLock(hCCD);


for ( j = NY-1; j >= 0; j--)
{
for ( i = 0; i < NX; i++)
{
ULONG data = MAKEWORD(pCCD[0], pCCD[1]);                                //A/D 変換 16bit の時
//pixdata[i][j] = data;
pixdata[j][i] = data;
pCCD += 2;
}
}
::GlobalUnlock(hCCD);
::GlobalFree(hCCD);
printf( "Pixdata  %d %d %d¥n", pixdata[50][71], pixdata[50][72], pixdata[50][73] );
printf( "Pixdata  %d %d %d¥n", pixdata[51][71], pixdata[51][72], pixdata[51][73] );
printf( "Pixdata  %d %d %d¥n", pixdata[52][71], pixdata[52][72], pixdata[52][73] );

printf( "Write fits file... status %d¥n", writefitsimage( pixdata ) );

// convert to 8bit data
for( j = 0; j <NY; j++){
for( i = 0; i < NX; i++){
pixel[j][i] = (pixdata[j][i] - C_low) * 256 / (C_high - C_low);
if(pixel[j][i] < 0)
pixel[j][i] = 0;
else if(pixel[j][i] > 255)
pixel[j][i] = 255;
}
}

// remove skylevel(LOS) and get center of gravity(GX,GY)
Jushin(pixel, &GX, &GY, &LOS);

if(guid_GX != GX || guid_GY != GY){
dGX = GX - guid_GX;
dGY = GX - guid_GX;

pos_now(&ratel, &dectel, &nsrot);
dtheta = (nsrot - orig_nsrot) * deg2rad;
```

```
        rotate_dGX = dGX * cos(dtheta) - dGY * sin(dtheta);
        rotate_dGY = dGX * sin(dtheta) + dGY * cos(dtheta);

        arcsec_dra = rotate_dGX * SF;
        arcsec_ddec = rotate_dGY * SF;
        SendTelescopeOffset(arcsec_dra, arcsec_ddec);
}
}
*/
}



//
// Initializing Socket communication
//
int SockInit( void ){
        WSADATA wsa;
        int ret;
        if( (ret=WSAStartup(MAKEWORD(1,1), &wsa))){
                char buf[80];
                sprintf( buf, "%d is the err", ret );
                printf("%s¥n", buf );
                exit( -1 );
        }
        return FALSE;
}

//
// Socket Connection (for Client)
//
int SockConnect( void ){
        SOCKADDR_IN cl_sin;

        sock = socket( AF_INET, SOCK_STREAM, IPPROTO_TCP );
        if( sock==INVALID_SOCKET){
                printf( "Socket() failed" );
                return TRUE;
        }
        memset( &cl_sin, 0x00, sizeof( cl_sin ) );
        cl_sin.sin_family = AF_INET;
        cl_sin.sin_port = htons( PORT );
        cl_sin.sin_addr.s_addr = inet_addr( HOST_NAME );

        if( connect( sock, (LPSOCKADDR)&cl_sin, sizeof(cl_sin))==SOCKET_ERROR ){
                if( WSAGetLastError()!=WSAEWOULDBLOCK){
                        closesocket( sock );
                        sock=INVALID_SOCKET;
                        printf( "connect() failed" );
                        return TRUE;
                }
        }
        return FALSE;
}

//
// Get current ra, dec, offset(ra, dec, nsrot)
//
double GetTelValue(TELEPARAM *teldata){
        char wbuf[1024], rbuf[1024];
        memset( wbuf, 0, sizeof( wbuf ) );
        memset( rbuf, 0, sizeof( rbuf ) );

        // R_A_OBJ, DEC_OBJ, OFF1_R_A, OFF1_DEC, OFF1_ROT2
        sprintf( wbuf, "A %03d %03d %03d %03d %03d¥n", 38, 39, 50, 51, 306 );
        if( send( sock, wbuf, strlen(wbuf), 0 )==SOCKET_ERROR ){
                printf( "Sending Socket connection failed." );
        }
        Sleep( 100 );
        if( recv( sock, rbuf, 1024, 0 ) <= 0 ){
                printf( "Receiving null data in Socket Connection." );
        }

        sscanf(rbuf, "A %lf %lf %lf %lf %lf¥n", &teldata->obj_ra, &teldata->obj_dec, &teldata->offset_ra, &teldata->offset_dec, &teldata->nsrot);

        return FALSE;

}

int TGetValueInt( int command ){
        char wbuf[1024], rbuf[1024];
        memset( wbuf, 0, sizeof( wbuf ) );
        memset( rbuf, 0, sizeof( rbuf ) );
```

```
        sprintf( wbuf, "A %03d¥n", command );
        if( send( sock, wbuf, strlen(wbuf), 0 )==SOCKET_ERROR ){
                printf( "Sending Socket connection failed.¥n" );
        }
        Sleep( 100 );
        if( recv( sock, rbuf, 1024, 0 ) <= 0 ){
                printf( "Receiving null data in Socket Connection.¥n" );
        }

        return atoi( &rbuf[2] );

}

int TGetOffset( TELSTAT *telstat ){

        char wbuf[1024], rbuf[1024];
        memset( wbuf, 0, sizeof( wbuf ) );
        memset( rbuf, 0, sizeof( wbuf ) );

        sprintf( wbuf, "A %03d %03d %03d %03d %03d %03d¥n",¥
                        50, 51, 52, 53, 54, 306 );
        if( send( sock, wbuf, strlen(wbuf), 0 )==SOCKET_ERROR ){
                printf( "Sending Socket connection failed." );
        }
        Sleep( 100 );
        if( recv( sock, rbuf, 1024, 0 ) <= 0 ){
                printf( "Receiving null data in Socket Connection." );
        }
        //AfxMessageBox( rbuf );
        sscanf( rbuf, "A %lf %lf %lf %lf %lf %lf", &telstat->ra, &telstat->dec,¥
                        &telstat->azi, &telstat->al, &telstat->ir1, &telstat->ir2 );


        free(telstat);

        return FALSE;
}

int TOffset( const TELSTAT *telstat ){
        char wbuf[1024], rbuf[1024];
        memset( wbuf, 0, sizeof( wbuf ) );
        memset( rbuf, 0, sizeof( wbuf ) );

        sprintf( wbuf, "P %.1lf %.1lf %.1lf %.1lf %.1lf %.1lf¥n",¥
                        telstat->ra, telstat->dec, telstat->ir1, telstat->ir2,¥
                        telstat->azi, telstat->al );

        if( send( sock, wbuf, strlen(wbuf), 0 )==SOCKET_ERROR ){
                printf( "Sending Socket connection failed." );
        }
        Sleep( 100 );
        if( recv( sock, rbuf, 1024, 0 ) <= 0 ){
                printf( "Receiving null data in Socket Connection." );
        }

        return FALSE;
}

//
// Move Telescope (sending offset)
//
int SendTelescopeOffset( double dalp, double ddel ){
        TELSTAT *telstat;
        telstat = (TELSTAT *)malloc( sizeof( TELSTAT ) );
        memset( (TELSTAT *)telstat, 0.0, sizeof( TELSTAT ) );

        TGetOffset( telstat );
        telstat->ra = telstat->ra - dalp;
        telstat->dec = telstat->dec + ddel;
        telstat->ir1 = telstat->ir1 * 3600;
        telstat->ir2 = telstat->ir2 * 3600;

        TOffset( telstat );
        free( telstat );

        return FALSE;
}

//
// main function
//
int _tmain(int argc, _TCHAR* argv[])
```

```
{
        // CCD Initiallize
        int temp_ccd, temp_body;

        printf( "CCDOpenDevice     : %s¥n", CCDOpenDevice() );

        if(CCDControllerReset() == TRUE)
                printf( "CCDControllerReset: TRUE¥n" );
        else
                printf( "CCDControllerReset: FALSE¥n" );

        /*---- the following line was deleted 2011-01-31 M.Yoshida
          printf( "CCDControllerReset: %d¥n", CCDControllerReset() );
          ----*/
        printf( "CCDGetInfomation  : %s¥n", CCDGetInfomation(NULL) );

        CCDGetTemperature( 0, &temp_ccd );
        CCDGetTemperature( 1, &temp_body );
        printf( "Temperature  CCD : %+.1f   Body : %+.1f¥n", temp_ccd/10., temp_body/10. );
        printf( "CCD device opened.¥n" );

        // Stage Inittiallize
        ALK_EnvironmentInfo_Tool(&AlkResult);
        XAxiOpen();
        YAxiOpen();
        printf( "Motor driver opened.¥n" );

        OriginDrive();

        //      SockInit();
        //      SockConnect();
        printf( "Circget¥n" );
        circget();

        detgstar();

        printf( "Start autoguide¥n" );
        AutoGuide();


        XAxiClose();
        YAxiClose();

        //      closesocket( sock );
        //      sock=INVALID_SOCKET;

        printf( "Now sleeping...¥n" );
        Sleep( 10000 );
        return 0;
}
```

謝辞

本研究にあたり，川端先生にはC言語や論文の書き方など多大なご指導をいただきました．心より感謝いたします．私の仕事が遅く，大変にご迷惑をお掛けしてしまいました．またソフトウェア作成に関しアドバイスをくださった，吉田先生，深沢先生，平木くん，伊藤くん，小松くん，プログラムテストに付き合ってくださった笹田くん，先本くん，山中さん，奥嶋さんにも心より感謝いたします．

参考文献

[1] 平木一至 「広島大学かなた望遠鏡用自動追尾システムの開発」（広島大学 卒業論文, 2008）

[2] 加藤大輔「南アフリカ 1.4m 望遠鏡の建設と評価」（名古屋大学大学院 修士論文, 2001）

[3] 柴田望洋 「明解 C 言語 入門編」（ソフトバンククリエイティブ，2004）

[4] 「AL シリーズ USB マスターユニット CB-23/USB 取扱い説明書」（Melec）

[5] 「AL シリーズ汎用入出力 CB-34/IO 取扱い説明書」（Melec）

[6] 「ステッピングモータコントローラドライバ CD-773/ADB5331A 取扱い説明書」（Melec）

[7] 「STEPPING & SERVO MOTER CONTROLLER'S OPTION MPL-28/ALUSBWXP 取扱説明書（デバイスドライバ AL MCC05 ユニット編）」（Melec）